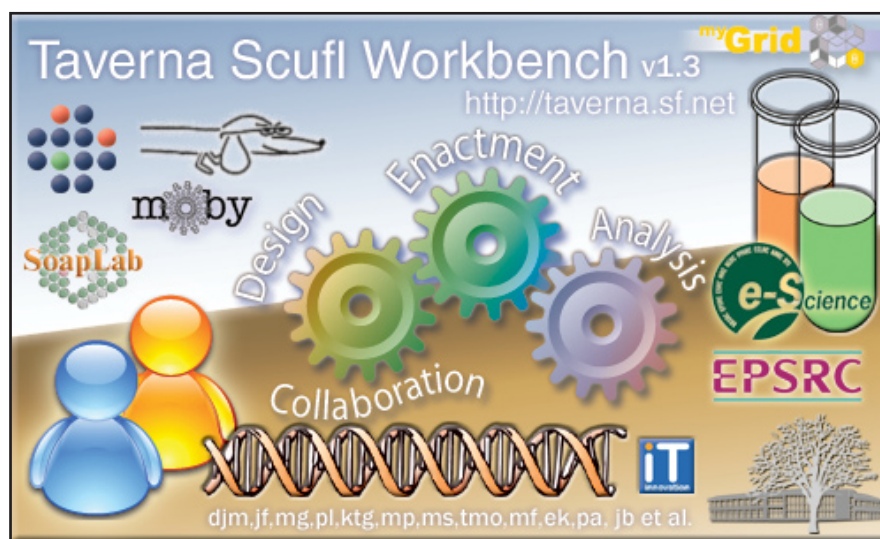


An Introduction to Taverna 1.3

The Taverna workbench software allows the construction of complex in silico experiments in the form of workflows, with a particular focus on the bioinformatics domain. We recognize, however, that in spite of much work on the user interface and tooling the task of creating an in silico experiment is inherently complex and the tools we provide are similarly so. This document contains a set of tutorials intended to lessen the initial learning curve but is not intended as a complete guide to all features in the current software. You are free to copy, modify and redistribute this document as you see fit and we welcome any suggestions aimed at enhancing its future utility for training courses or personal study.



Contents

Exercise 1 – Downloading, installing and running Taverna 1.3	2
Step 1 – Download the appropriate Taverna workbench archive.	
Step 2 – install prerequisites (Linux & Windows only)	
Step 3 – Launch the Taverna workbench	
Exercise 2 – Browsing for and invoking a single service	3
Step 1 – Launch Taverna 1.3	
Step 2 – Find the services panel	
Step 3 – Using the quick search	
Step 4 – Prepare to invoke the single service	
Step 5 – Enter input data	
Step 6 – Invoke the service	
Step 7 – Examine results	
Step 8 – Save results (optional)	
Step 9 – View log (optional)	
Exercise 3 – Loading, exploring and invoking an example workflow	5
Step 1 – The Advanced Model Explorer window	
Step 2 – Loading an example workflow	
Step 3 – The diagram views	
Step 4 – Workflow descriptions in the AME	
Step 5 – Invoking the workflow	
Step 6 – Workflow status	

Step 7 – More result browsing

Exercise 4 – Creating a very simple workflow from scratch

7

Step 1 – Clearing the current workflow

Step 2 - Workflow inputs and outputs

Step 3 – Adding a single sequence fetch component

Step 4 – Connecting everything together

Step 5 – Describing the input

Step 6 – Enacting the workflow and seeing the results

Exercise 5 - Sharing your workflow and reusing existing workflows

8

Step 1 - Publishing a workflow online

Step 2 - Loading a workflow from a URL

Step 3 - Loading workflows into the services panel

Step 4 - Workflows in the services tree

Step 5 - Importing a workflow from the services tree

Step 6 - Collecting a set of workflows

Where to go next?

10

Alternative rendering of results

Iteration control

Fault tolerance

Inline scripting

Biomart data warehouse queries

Exercise 1 – Downloading, installing and running Taverna 1.3

This (hopefully very short!) exercise should end with a working installation of Taverna 1.3 on your laptop or workstation. This is rather important to get right – you'll find the other exercises somewhat harder if you don't have the software working.

Step 1 – Download the appropriate Taverna workbench archive.

The current version of Taverna is always available on the web at <http://taverna.sf.net>, as are archives of all previous versions. The correct download depends on your operating system

If you are using either a modern version of Windows (Win2k or WinXP, with XP preferred) or any form of linux, solaris etc. you should download the workbench zip file (which will have a name like 'taverna-workbench-1.3.zip'). We do not support older versions of Microsoft's operating systems, if you are running Win95, 98, ME or similar please do yourselves a favour and upgrade!

On Windows you should use WinZip to extract the .zip file to somewhere on your hard drive. Although WinXP does include an archive utility it has proved to be painfully slow, slow enough in fact that it's actually faster to download WinZip, install it and use it to open the archive than it would be to use the native extractor. You should end up with a directory in which there are several .bat files including a 'runme.bat'.

On linux and other UNIX variants you should use whatever archive software is installed, normally 'unzip' to open the archive and put the contents somewhere you have write access to (Taverna doesn't care where it is placed).

If you are using Mac OSX you should download the .dmg workbench file. Note that although Taverna works on OSX there may be components that run sub-optimally due to the relatively low performance of the Apple Java Virtual Machine (JVM) in comparison to the Windows, Solaris and Linux versions. Apple do, however, now include a JVM with OSX so you shouldn't need to install anything extra.

When the .dmg file is downloaded to your desktop it may automatically be opened as a disk image (if not then simply double click to open it manually). This will reveal two icons – the Taverna cog icon and another application called GraphViz. The latter is used by Taverna to generate workflow diagrams and other uses.

You cannot run Taverna from the disk image (we don't know why) so you'll have to copy both applications to the same place somewhere on your hard disc. It is essential that both applications are copied and that you do not change the name of the GraphViz app!

Step 2 – install prerequisites (Linux & Windows only)

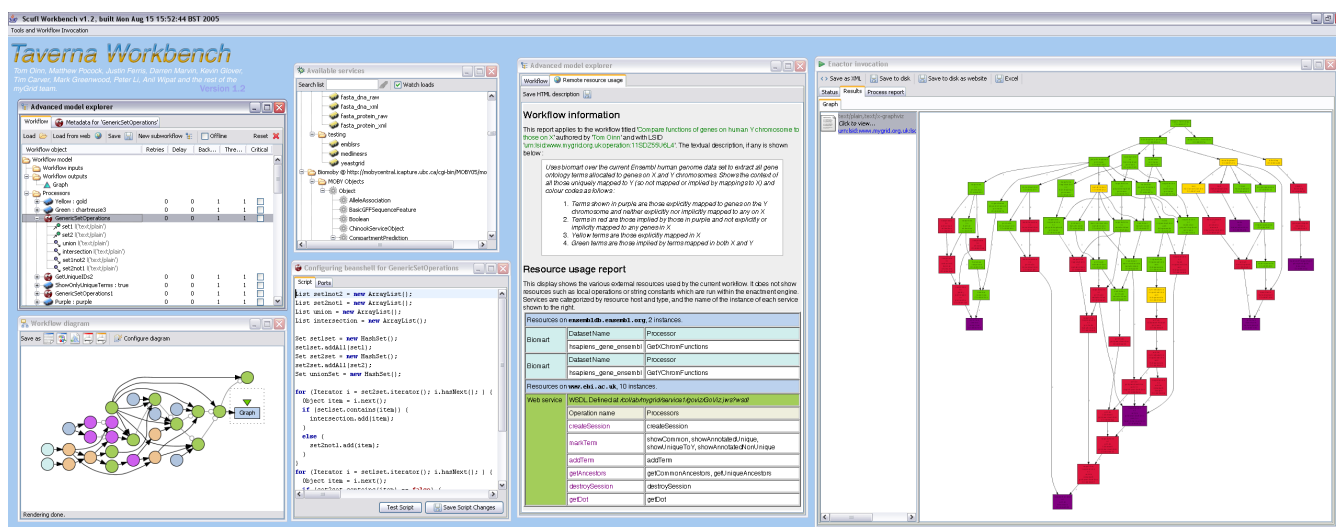
Taverna has an additional dependency on a package called GraphViz from the former AT&T research labs. On Windows we include a copy of the appropriate binaries and you shouldn't need to install anything, on linux however you will have to install the appropriate version of graphviz for your platform if you do not already have it. The details of software installation will vary from platform to platform, we are aware that there is a graphviz RPM for distributions based on Redhat Linux and that this provides the required functionality.

On both Windows and Linux you will also need to ensure you have a modern Java Runtime Environment (JRE) or Java Software Development Kit (SDK). These can be downloaded from <http://java.sun.com> but the odds are high you already have a suitable installation (note that Windows does not include one by default though). Versions that we believe work with Taverna are 1.4.2 and 1.5, if you have either version you should be fine.

Step 3 – Launch the Taverna workbench

Launching Taverna is simple, the mechanism dependant on your operating system:

- » Windows – run the 'runme.bat' file.
- » OSX – double click on the Taverna cog icon.
- » Linux – run the 'runme.sh' file.



In all cases you should see the workbench splash screen appear, followed by a large window with several other windows contained within it. This is the Taverna workbench and if you've got this far you now have a copy of it installed and running on your machine ready for the next exercise (the image above shows a fully populated workbench after some use, it won't look quite like this at first!)

Exercise 2 – Browsing for and invoking a single service

Before moving onto workflows we will use Taverna's list of available services (the tools that you will be linking together later) to run a single service. This introduces the concept of setting initial input data, monitoring a running process and viewing the results, all of which are identical when applied to entire workflows.

Step 1 – Launch Taverna 1.3

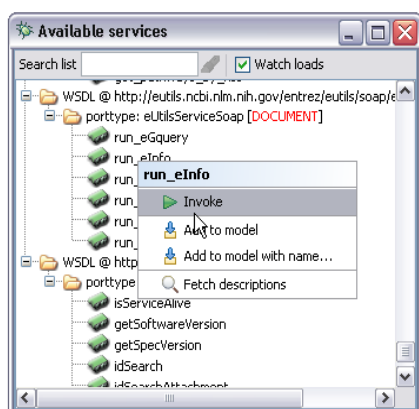
If you are doing this immediately following the first exercise you already have Taverna running, if not you should refer to exercise 1 step 3.

Step 2 – Find the services panel

The services panel is the window containing a tree of available services, by default on the right hand side of the main Taverna window. Taverna loads a set of default services when it starts up (this is controlled by configuration in the `conf/mygrid.properties` file if you need to change the starting set), these may take some time to load and you should wait until the tree starts expanding itself so all nodes are visible. You should see that services can be part of categories and that the leaf nodes (nodes with no children in the tree) are variously coloured, we'll get to what the colours mean later.

Step 3 – Using the quick search

There are a significant number of services, probably too many to simply scroll through and expect to find anything useful. Taverna includes several search mechanisms including some highly sophisticated ones but in this case we'll be using the simple text search box at the top of the services panel. Enter the text 'GODBGetNameByID' into the search box and hit return or enter. You should see the list of available services contract – what has happened is that all nodes that have text which doesn't match are collapsed and those that do are highlighted red and expanded. Scroll down the panel to find the single node with red text, it should have a dark blue chip icon. The colour in this case tells you that the service is based on a system called Seqhound and if you scroll up to find the immediate parent it tells you that the service is running in Canada at seqhound.blueprint.org (okay, so you would have to know that this host is in Canada but you can trust me for now!)



Step 4 – Prepare to invoke the single service

Scroll back down to the service you just located and right click on either the icon or the red text (for the purposes of this tutorial when you are asked to right click you should perform whatever combination of key and mouse presses are required on your platform and windowing system to bring up a context menu, I'm writing this on a WinXP machine but all platforms have similar functionality). You should see a menu containing various options, top of which is 'invoke'. Select this option and you should see a new window appear, you will use this window to set the input data for the service to work on.

Step 5 – Enter input data

The new window, entitled 'Run Workflow' (run workflow rather than run service because what you have just done is created a very simple workflow with a single service in!) should contain a very simple tree with a single leaf node called 'goid'. This corresponds to the single named input for this service. The service you located consumes a Gene Ontology (GO) identifier and returns the name of the term corresponding to that identifier. It accepts IDs in the form of a decimal numeral without leading zeroes.

Select the 'goid' node in the tree – the window should split into two panes; the tree on the left and a new pane on the right. This new pane along with the now enabled button in the toolbar allows you to define the inputs (or input, in this case) for the process. Click on the 'New Input' toolbar button, you should see a child node of 'goid' appear in the tree on the left and a single text string appear in the pane on the right. Edit the string 'Some input data goes here' in the right hand panel to be '7601' (or any other GO id of your choice).

Step 6 – Invoke the service

Once you have your input data defined you are ready to run the service, click on the 'Run Workflow' button located on the bottom right of the panel. This will lead to another window being opened containing the status of the process you just launched.

Step 7 – Examine results

The new window should very briefly show the progress of your workflow before jumping almost immediately to the results browser. You will see a single tab ‘result’ for the output from the service and a similar view to that used to enter the input data; a tree on the left (containing only one node so hardly a tree) and detail on the right which initially contains some simple instructions on how to use the result browser. As the returned value, ‘visual perception’ if you used the default input in step 5, is very short there is no need to select it, you can read the result directly from the tree. Were this a longer output, a BLAST report for example, you would select the document in the tree and read the full version in the right hand panel.

Step 8 – Save results (optional)

In this case the result is hardly worth saving, however, Taverna provides several mechanisms for storing your workflow outputs to disk. You can right click on any individual data item to save that particular item, or use the buttons in the toolbar to save all results to disk or Excel (only textual results are saved to Excel but it can be ideal for structured output such as lists of scores or ids).

Step 9 – View log (optional)

Every time Taverna runs a workflow, even a simple one such as this, a log is created of exactly what happened to each operation in the workflow. In most cases you will never need to examine this but if you wish to see a report of the invocation you’ve just performed you can select the ‘Process report’ tab in the results window, you will see a tree of XML containing a list of all the processors (Taverna’s term for a service within a workflow) each of which will contain a list of events from the initial scheduling of the process up to completion or failure. This is generally only essential if the workflow has failed for some reason or during testing.

Exercise 3 – Loading, exploring and invoking an example workflow

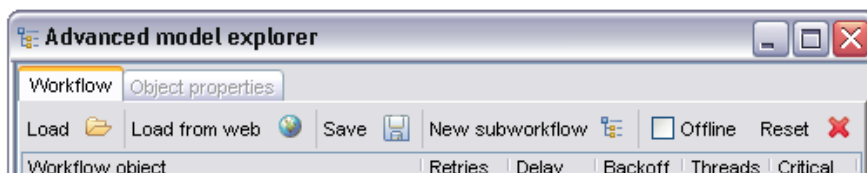
This exercise will demonstrate the use of the workflow diagram and model explorer views as well as a new output type and the monitoring of a more complex invocation.

Step 1 – The Advanced Model Explorer window

This exercise will make use of the two windows that you didn’t use in exercise 2. First of these is the Advanced Model Explorer (AME). This presents a view of everything in the current workflow and allows you to load new workflows, modify them or create them from scratch and to save your changes. In this case we’ll run one of the examples.

Step 2 – Loading an example workflow

Taverna includes a set of example workflows of varying complexity in the ‘examples’ folder of the workbench distribution. Locate the AME and click the ‘Load’ icon – this will bring up a platform standard file chooser, you can use this to navigate to wherever Taverna’s examples directory is located. You’re looking for a file called ‘ShowGeneOntologyContext.xml’, a workflow definition in XML form. Once you have this file selected click ‘Open’ in the file dialogue and, after a brief delay, you should see the AME and diagram views populated.



Step 3 – The diagram views

One of the key advantages of all workflow systems over conventional scripting is the comparative ease with which a user can comprehend what a workflow does. Key to this is the graphical representation. In the default view you can consider data flowing in from the top of the diagram, running through a set of processes (the

coloured boxes) and eventually leaving the workflow at the bottom where it is presented to the user. The arrows between the boxes indicate where data is moved from the output of one process to the input of the next, lines with circles at their ends denote that one process should only start once another has finished.

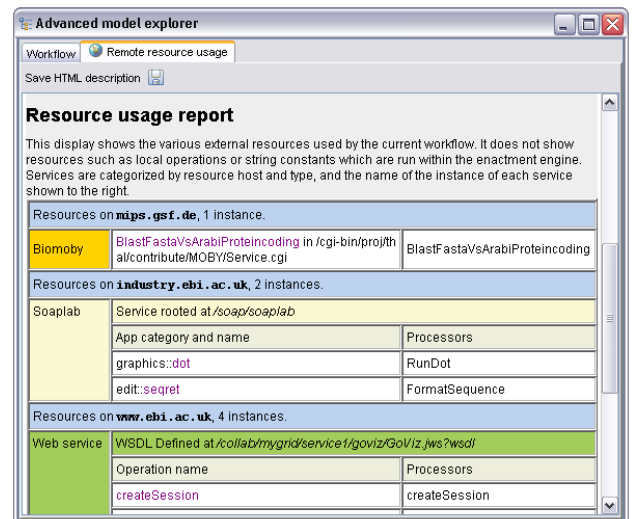
By default Taverna's diagram view shows no ports (the named inputs and outputs used by each process – the 'goid' input in exercise 2 was an input port), lays the diagram out from top to bottom, shows things even if it thinks you might not want to see them and doesn't attempt to show any information about the type of data moving between each entity. All of these settings may be changed by clicking on the 'Configure diagram' button and select or deselecting options in the drop down menu that appears. The best way to become familiar with this is to play with it!

For now the best way to see this particular workflow is to leave all settings unchanged but unset 'show boring entities' – this should clean up the workflow diagram substantially.

You can navigate around the diagram, zooming and panning, by holding down (on Windows) the 'shift' key and dragging with either left or right mouse buttons, this makes the larger workflow diagrams considerably more tractable! To reset the workflow to fit the current window size you can open the diagram configuration menu and select the current view mode (no ports, all ports etc) again, the diagram will be rendered to fit the panel.

Step 4 – Workflow descriptions in the AME

The AME is a highly complex component, this exercise only considers a subset of its functions. One thing that is particularly important when loading a new workflow is the ability to obtain documentation! In Taverna's case workflows may be annotated with various textual metadata such as authorship, description, title etc and may also show autogenerated information in the form of a report on where exactly the services used by the workflow are being run. To see this information you need to select the 'Processors' node in the tree. A new tab should become visible at the top of the AME with the text 'Remote resource usage', select this tab and you will see a description of the workflow along with a report telling you that the workflow is using web service operations located at the European Bioinformatics Institute (www.ebi.ac.uk).



Step 5 – Invoking the workflow

Having read the description and seen the diagram you should have at least some idea of what the workflow will do (but if not don't worry, it's not overly important). The next stage is to run the workflow, this is done by opening the menu 'Tools and Workflow Invocation' at the very top of the main workbench window (Windows and Linux) or in main menu bar at the top of the screen (OSX). Once you've selected this you will see an input form very similar to the one which appeared in exercise 2. Using the same methods you should run the workflow, again on term 7601 but first note the additional information shown when you select the 'termID' input – in particular you'll see that the ID is required in a different form!

Step 6 – Workflow status

When invoking a single service in exercise 2 the workflow status display jumped almost immediately to the results browser. In this case, however, the workflow takes a non negligible time to complete and will show you detailed progress information as it goes. The status display consists of two portions – the upper half of the window contains a tabular view of all processors along with detailed status whereas the lower half contains a graphical view of the progress, you should see the nodes in the workflow diagram change in the status view as they move from scheduled through running to completed status. When all nodes have completed you will again be jumped to the result browser.

Some parts of this workflow involve looping over a set of inputs and performing the same operation on each input to produce a set of outputs. Where this is happening you will see a progress bar overlaid on the graphical status and the detail message ‘Iteration number = ...’ in the tabular view to inform you how far through the loop the process is at that time.

Step 7 – More result browsing

The result this time is more interesting than a simple piece of text, as the workflow description suggests the results shows the term you selected in the context of the rest of the GO structure. The same navigation controls apply to this diagram as to the workflow diagram used in step 3, you should be able to zoom and pan around the image and note the colouring that the workflow has created.

You’ve now run a considerably more complex workflow, monitored its status and explored some of the workflow properties using the AME and diagram views, next we will create a simple workflow from scratch.

Exercise 4 – Creating a very simple workflow from scratch

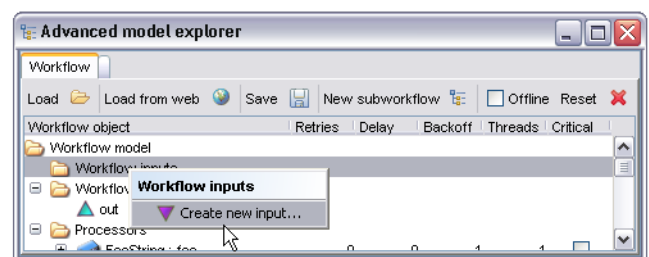
This exercise walks through the process of creating an extremely simple workflow from scratch using a single operation from the EMBOSS tool suite.

Step 1 – Clearing the current workflow

By default Taverna will always add to the current workflow so to create a new one you must explicitly tell the system to clear the current state. You can do this by clicking on the Reset icon (a red cross) in the AME, if you have followed this literally from exercise 3 you will first need to select the ‘Workflow’ tab to go back to the normal AME view.

Step 2 - Workflow inputs and outputs

First things first - the workflow needs to have an input, in this case the id of a sequence to fetch. You can create a new workflow input by right clicking on the ‘Workflow inputs’ node in the Advanced Model Explorer and selecting ‘Create new input’. This will then ask you for a name for the input, you can change this later but for now use ‘sequenceID’. Click on ‘OK’ and you should see your new input appear in the Advanced Model Explorer and Workflow Diagram windows.



Similarly, the workflow will need an output. Follow an equivalent process but this time clicking on the ‘Workflow outputs’ node and using the name ‘sequence’ to create an output. Again, you should see the output appear in the two windows.

Step 3 – Adding a single sequence fetch component

Now that the workflow has an input and output it needs something to fit between to actually do the work. If you’re familiar with the EMBOSS tool suite you might be aware that there’s a program called ‘seqret’ which can do exactly this, fetch a sequence from an ID. Fortunately the default services available from Taverna include a system called Soaplab at the EBI which contains all EMBOSS tools, including seqret. Go to the ‘Available Services’ window and either scroll down until you find it or, more sensibly, enter ‘seqret’ into the search box at the top and hit return as in exercise 2.

To add the operation to the workflow you can drag the 'seqret' service from the 'Available Services' window into the 'Advanced Model Explorer' window, dropping the service into the empty space on the right of this window. You should hopefully see a new entry under the 'Processors' node called 'seqret' with an array of other child items hanging off it - these are the available inputs and outputs to and from the newly created process.

Step 4 – Connecting everything together

You should see the workflow input, processor and workflow output in the 'Workflow Diagram' window, but they're not linked together yet. To link them, you need to right click on the 'sequenceID' node in the 'Advanced Model Explorer' and select the 'seqret' child menu (under the 'Link sequenceID to...' item). This child menu will show you all the inputs in the seqret tool to which you can link your workflow input. The exact details of all these inputs will depend on the processor, in this case you want to link the sequenceID workflow input to the 'sequence_usa' processor input. Select the 'sequence_usa' from the child menu and you should see the link appear in the workflow diagram.

The remaining link in the workflow is from the output of the processor to the workflow output. Conceptually we always link from the source of the information to its destination, so you'll need to expand the 'seqret' node in the Advanced Model Explorer (if it isn't already expanded) and scroll down to see the available outputs, denoted by purple circles with outgoing arrows. There are two outputs from this processor, 'report' and 'outseq', the one we want to use is the 'outseq'. Similarly to the first link, right click on the 'outseq' node and follow the 'Workflow outputs' child menu. There is only one workflow output so select it and you should observe the now completed workflow in the 'Workflow Diagram' window.

Step 5 – Describing the input

You can add descriptive information to the workflow inputs - this information is useful to guide potential users of your new workflow. Select the 'sequenceID' node by left clicking on it in the 'Advanced Model Explorer'. You should observe a tab at the top of the window with a purple icon and the text 'Metadata for 'sequenceID''. Select this tab by clicking on it.

This is the metadata editor. For now we're only going to look at the free text description, the ontology and mime type tabs are too complex for this introduction. Select the 'Description' tab, enter some description of the input (i.e. a quick paragraph explaining that this input should hold the sequence ID to fetch and perhaps an example value - 'swallid:ops2_*' is a reasonable one), click the 'Update' button when you're done and jump back to the 'Workflow' tab to get back to the default view of the workflow.

Step 6 – Enacting the workflow and seeing the results

You should now be able to run the workflow just as in exercise 3, try entering your example 'swallid:ops2_*' as the input (without the quotes) and you should see the status display (briefly) followed by the results which, all being well, should contain the fetched sequences in fasta format. If this worked then great, you've created a simple workflow from scratch and enacted it.

Exercise 5 - Sharing your workflow and reusing existing workflows

This exercise revolves around the idea that collaboration should be a fundamental part of scientific work. Taverna facilitates collaboration by allowing workflows to be easily shared online and by providing mechanisms whereby workflows can either be treated as components themselves or where components within those workflow may be selectively cloned out and reused

Step 1 - Publishing a workflow online

This isn't really a Taverna step at all, publishing a workflow in this context simply consists of making it available to the world via the web. In a trivial case you can save the XML definition from the AME (click on the Save icon) and move the workflow XML file to, for example, the public_html directory on a standard UNIX account. If you have access to a more organized Workflow Repository (as we will hopefully for Nettab) you can use that mechanism instead.

Step 2 - Loading a workflow from a URL

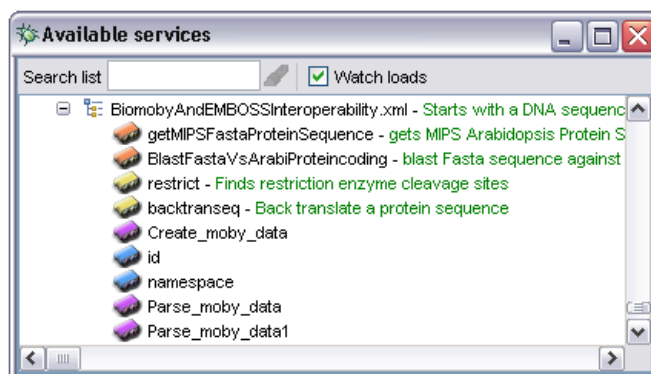
The AME allows you to load an entire workflow from a URL. Choose a workflow that someone else has put online and obtain the URL from them. Click on the 'Load from web' icon and enter this URL into the dialogue box that opens - you should find the workflow definition loads exactly as if you'd had direct access to the file. Obviously this is a very simplistic mechanism but can be useful.

Step 3 - Loading workflows into the services panel

The services panel may be populated with workflow definitions. These can be added in two ways, either explicitly or by initiating a web crawler which will proactively hunt within a website for workflow definitions and load any that have been found. Find the services panel (as used in exercise 2) and scroll all the way to the top. Right click on the root node of the tree and you should be presented with a menu allowing you to add new content to the services tree. To add a single workflow as a service (for example the one you found in step 2 of this exercise) you should select the 'Add new workflow scavenger...' option, as before this will prompt you for the URL of a workflow definition; enter the same URL and scroll down to the bottom of the services tree.

Step 4 - Workflows in the services tree

You should see the workflow definition you just loaded appear as a new node in the tree. As with any other kind of service you can drag from here into your AME to load the entire workflow as a single process. You will notice, however, that the workflow itself has child nodes in the tree; these correspond to the processors within that workflow. See a service in someone else's workflow that you want to use but don't know where it came from? Simply load their workflow into the services panel, open the workflow node to find the service you want to use in your workflow and drag it in as normal. This is particularly useful for service types which require significant configuration before they will function such as Beanshell scripts and Biomart data warehouse queries



Step 5 - Importing a workflow from the services tree

If you right click on the workflow node you added in step 3 you'll see an option that only appears on workflow nodes and not on any other services - 'Import workflow'. If you select this the workflow will be loaded as is rather than as a single component. This option also allows you to specify a prefix which will be applied to everything within that workflow before it is added, this is because some output names (for example 'output') are commonly used by workflow designers and you would be unable to add such workflows without renaming everything first as Taverna does not allow duplicate names. If you leave the prefix blank no renaming will be done.

Step 6 - Collecting a set of workflows

As mentioned in step 3 it is possible to initiate a web crawl for workflows. This will also find some other service descriptions (such as SOAP WSDL definitions) but the primary purpose is to locate workflows. Right click on the top level node in the services tree as in step 3 and select 'Collect scavengers from web...'. This

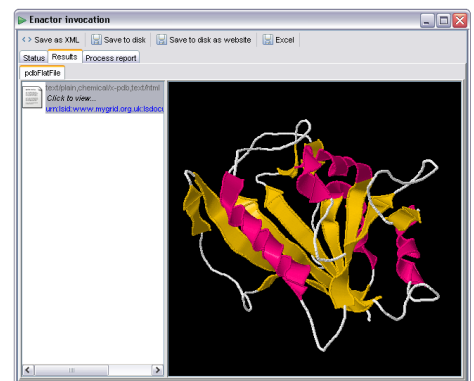
will then give you a dialogue box into which you should enter the root location to start the web crawl from. This works particularly well if you have a standard Apache like web server set up to allow directory listing and have all your workflows in a directory within your web space. Assuming you have a suitable target to search you should find a new node in the services panel with the workflows the search located as immediate children.

Where to go next?

Taverna and its associated components are capable of creating highly sophisticated analysis applications, some of which are presented as posters at this meeting. This introduction has only scratched the surface of what the system is capable of, in particular there are several areas that you might wish to explore further with the help of the user manual.

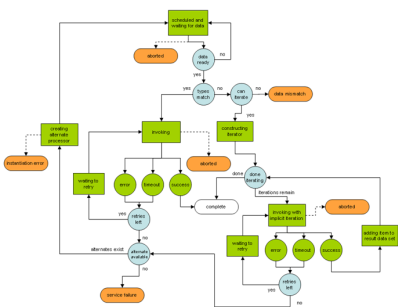
Alternative rendering of results

In exercise 4 we added some descriptive metadata to the workflow input to provide a user with a hint as to the form of IDs the workflow expected. Similarly, metadata can be applied to workflow outputs in order to drive the selection of an appropriate renderer for the data produced. For example, the workflow loaded in exercise 3 had an output which was annotated as being of type 'text/x-graphviz' which in turn allowed Taverna to create the interactive view of the Gene Ontology graph rather than simply presenting the text that generated it. The manual has details of some of the MIME types you can use, the most common being the 'image/*' forms to denote graphical data but we also have some more sophisticated ones allowing live display of 3d structures and similar specific operations.



Iteration control

The exact mechanism by which Taverna loops over sets of data when you create a mismatch between what a process expects and what it is given is highly configurable - in a sense this iteration strategy mechanism is at the heart of the power of most workflows and is essential to master for advanced use.

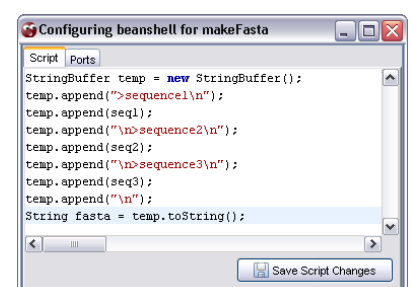


Fault tolerance

Services, being located on other people's computers, are potential sources of failures. Taverna includes mechanisms to allow you to specify retry behaviour and failover to alternative services in these cases. Alternate processors also allow a form of exception handling within the workflow system, particularly valuable in conjunction with extensions such as the Interaction Service which allows a mostly automated process to include explicit interaction with an expert reviewer.

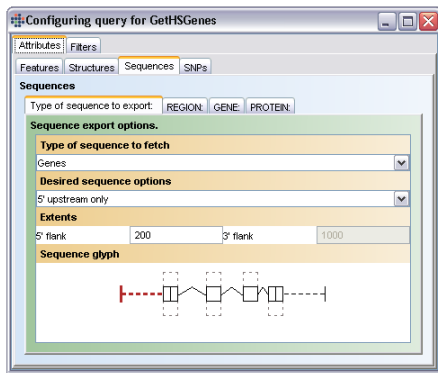
Inline scripting

You will have noticed that we had two forms of the same ID in these examples, one ID looked like '7601' and the other 'GO:0007601'. Clearly we could create a service that converted between these two representations but a full networked service is excessive and inefficient in this case. Taverna allows expert users to construct services from interpreted Java code using the Beanshell extension.

A screenshot of a dialog box titled 'Configuring beanshell for makeFasta'. The dialog has two tabs: 'Script' and 'Ports'. The 'Script' tab is active, showing a Java script for creating a FASTA file. The script is as follows:

```
StringBuffer temp = new StringBuffer();
temp.append(">sequence1\n");
temp.append(seq1);
temp.append("\n>sequence2\n");
temp.append(seq2);
temp.append("\n>sequence3\n");
temp.append(seq3);
temp.append("\n");
String fasta = temp.toString();
```

At the bottom of the dialog, there is a 'Save Script Changes' button.



Biomart data warehouse queries

Taverna has the ability to connect to the Biomart system, your workflows can therefore access the current known data from several critical public data sets including Ensembl, VEGA and DbSNP. The manual contains a detailed section on how to construct and configure such queries and the ‘CompareXandYFunctions’ workflow can be used as an example.

Conclusion

Taverna is an active project and we value your feedback. If you have any further questions and especially if you’re considering using Taverna as part of your ongoing scientific work we’d love to hear from you. The website at <http://taverna.sf.net> contains links to all our documentation and plans and includes a page where you can sign up to one of our two mailing lists, one for developers and those interested in the code and one for those more interested in the application of the technology to their work.

Similarly, we value all contributions to the code itself, Taverna is an open source project under the LGPL license and has benefited hugely from developer contributions from outside the core myGrid project team, we have designed the system to be easily extensible so there’s scope for code varying from a few lines for a new renderer up to serious hacking of the core architecture, again just let us know!

